



## Tema 2. Algoritmi: caracteristici, reprezentare, implementare

### Obiective

- să urmărești execuția algoritmilor pas cu pas
- să identifice valorile variabilelor la fiecare pas
- să creezi algoritmi alternativi

### Fișa de documentare 2.3. Programarea structurată (II)



#### Structura alternativă

Auzim în viața de zi cu zi afirmații de genul: **DACĂ** obțin note de promovare la toate examenele, **ATUNCI** voi lua diploma, **ALTFEL** trebuie să mai învăț.

Se remarcă trei cuvinte ce au un rol deosebit: **DACĂ**, **ATUNCI**, **ALTFEL**. Propoziția are trei componente și anume:

- condiție, transcrisă prin “obțin note de promovare la toate examenele”, condiție pe care o notăm cu c;
- acțiune transcrisă prin “ voi lua diploma”, notată cu p, acțiune asociată cu **ATUNCI**, adică se execută doar dacă “obțin note de promovare la toate examenele”;
- acțiune transcrisă prin “ trebuie să mai învăț”, notată cu q, acțiune asociată cu **ALTFEL**, adică se execută dacă NU “obțin note de promovare la toate examenele”;

Folosind notațiile făcute, afirmația se poate transcrie în pseudocod sau schemă logică.



Secvența de instrucțiuni se numește **structură alternativă** și se poate reprezenta și grafic. Structura alternativă admite și o formă particulară, caz în care avem o ramură vidă (adică nu se execută nici o operație):

```

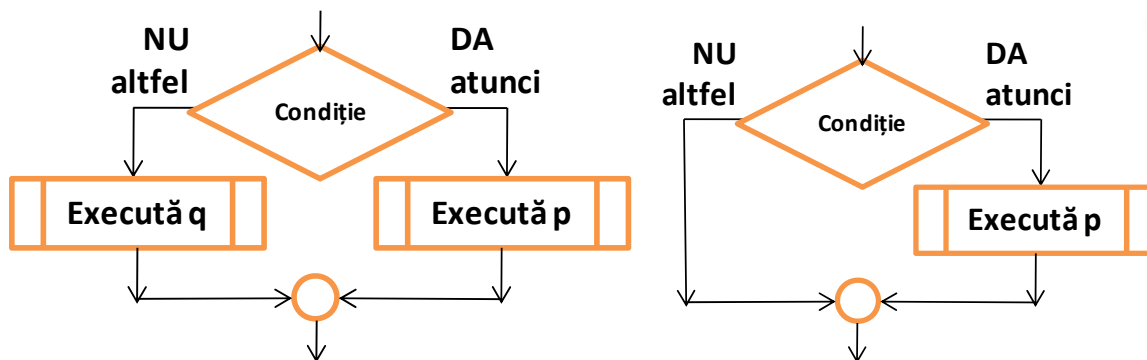
dacă c atunci
|   execută p
altfel
|   execută q
|

```

```

dacă c atunci
|   execută p
|   ■

```



Exista cazuri în care condiția poate fi mai complexă, de genul:

DACA obțin note de promovare la toate examenele ȘI toate notele sunt peste 9, ATUNCI voi putea beneficia de bursă, ALTFEL nu.

Notând prima condiție cu c1 (obțin note de promovare la toate examenele), cu c2 a doua condiție (toate notele sunt peste 9), cu p acțiunea „voi putea găsi un job cu salariu mai mare” și cu q acțiunea „salariul va fi mai mic”, se va folosi operatorul logic „and” iar condiția va fi compusă: „c1 and c2”.



### Observații



Atât ramura „ATUNCI” cât și „ALTFEL” permit executarea unei singure instrucțiuni. În cazul în care este necesară efectuarea mai multor instrucțiuni, acestea se grupează într-o singură instrucțiune compusă.



Uneori avem o instrucțiune de decizie subordonată unei alte instrucțiuni (de decizie sau de alt fel). Este important ca instrucțiunea subordonată să fie scrisă identat față de instrucțiunea care o subordonează. Acest mod de scriere nu este obligatoriu pentru funcționarea algoritmului însă face programele mai ușor de urmărit și de actualizat.



**APLICAȚII:** Prezentăm în continuare doi algoritmi alternativi importanți:



### Paritatea unui număr

Se introduce de la tastatură un număr întreg x. Să se testeze dacă numărul este par sau nu și să se afișeze un mesaj corespunzător.

**Exemplu:** dacă pentru x se citește valoarea **4123** se va afișa „Nu este par” iar pentru valoarea **588** se va afișa „Este par”.



```
x întreg //date de intrare
citește x
dacă x%2=0 atunci
|   scrie "Este par"
|altfel
|   scrie "Nu este par"
|
```



**Explicarea algoritmului:** Pentru a verifică dacă un număr este par trebuie să verificăm dacă restul împărțirii lui la 2 este „0”. În caz afirmativ rezulta ca numărul este par, altfel el este impar.



### Maximul între două numere

Se introduc de la tastatura două numere întregi x și y. să se afișeze numărul care este mai mare între cele două. În caz ca sunt egale, se va afișa un mesaj corespunzător.

**Exemplu:** dacă pentru x și y se citesc valorile **612 și 3129** se va afișa **“3129”** iar pentru valorile **58 și 58** se va afișa **“Numerele sunt egale”**.

```
x, y întregi //date de intrare
citește x, y
dacă x=y atunci
|   scrie "Numerele sunt egale"
|altfel
|   dacă x>y atunci
|   |   scrie x
|   |altfel
|   |   scrie y
|   |
```



**Explicarea algoritmului:** Pentru a afișa maximul între două numere le vom compara. Dacă cele două numere sunt egale vom afișa mesajul „Sunt egale”, altfel verificăm dacă  $x > y$ , situație în care vom afișa pe x, altfel vom afișa pe y.